Efficient Work Team Scheduling: Using Psychological Models of Knowledge Retention to Improve Code Writing Efficiency

Michael J Pelosi, Michael Scott Brown,
Henry Dirska, and Mir Mohammed Assadullah
University of Maryland University College
Largo, Maryland, United States

ABSTRACT

Development teams and programmers must retain critical information about their work during work intervals and gaps in order to improve future performance when work resumes. Despite time lapses, project managers want to maximize coding efficiency and effectiveness. By developing a mathematically justified, practically useful, and computationally tractable quantitative and cognitive model of learning and memory retention, this study establishes calculations designed to maximize scheduling payoff and optimize developer efficiency and effectiveness.

Keywords: WBS; scheduling; team; knowledge; retention.

1. INTRODUCTION

An interesting and classical quantitative law of cognitive psychology is that forgetting curves are well described by power functions (e.g., [1], [2], [3]). For example, Wixted and Ebbesen (1991) [3] and Wixted and Carpenter (2007) [4] show that diverse measures of forgetting various items such as words, faces, and nonsensical syllables can be well described as power functions of a retention interval.

In one seminal and classic article, Wickelgren (1974) [4] derived an equation that is robust in several respects, including in its ability to characterize the previously famous Ebbinghaus (1885, 1913) [5] memory savings function. With most typical conditions, the Wickelgren power law is reduced to:

$$m = \lambda(1 + \beta t)^{-\psi} \quad (1)$$

where m is memory strength, and t is time (i.e., the retention interval). The equation has three parameters: λ is the state of long-term memory at $t = 0$ (i.e., the degree of learning), ψ is the rate of forgetting, and β is a scaling parameter.

Wixted (2004) [6] also substantiated that Equation 1 provides a very accurate description of forgetting data that have been averaged over many subjects. It therefore not only fits the data well in terms of the percentage of variance, a relatively weak test, but also accurately predicts where future points will fall as the retention interval increases, which is a relatively stronger test. As a result of these averaging of artifacts in the group data, a possible stronger tests would be to accurately predict the degree of forgetfulness for individual learning subjects. One practical problem with averaging the testing artifacts is that such data are usually quite noisy. However, the eight measured data points of the classic Ebbinghaus (1885, 1913) [5] savings function have offered one possible rare and well-known exception. Previous research work has shown that the Ebbinghaus data can be reasonably well characterized by a two parameter power function of the form:

$$m = \theta t^{-\psi} \quad (2)$$

This power function has been considered an approximation of Equation 1 (Anderson & Schooler, 1991 [1]; Wixted & Ebbesen, 1991 [3]). Although Equation 2 offers a much better fit of the savings function than many other two-parameter candidates, it is undefined at $t = 0$, which is theoretically unsatisfying and limits the equation's practical utility. As one example, it could not be used to estimate the degree of learning. Typical memory decay rates based on measured experimental results from Equation 2 are shown below in Figure 1.

2. RETENTION EFFECT

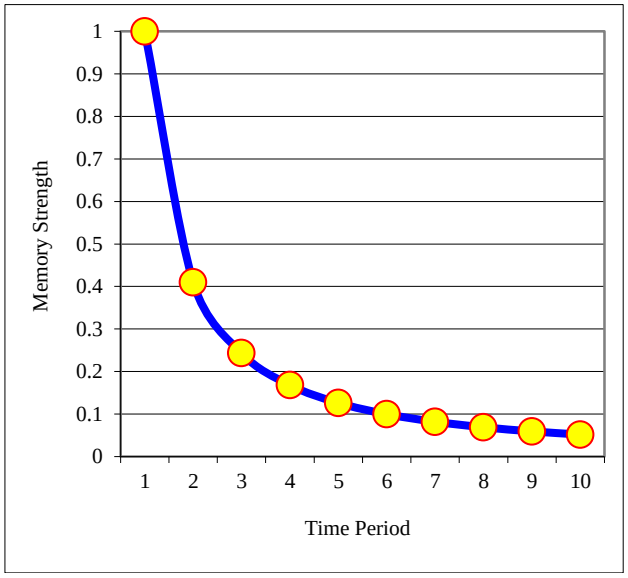

Figure 1. Typical retention decay rates (ψ = 1.286) from Donkin, 2012 [7].

Another study suggests that the probability that a previously learned memory item can be correctly recalled decreases at an exponential rate over time like radioactive decay. However, a so-called "spacing effect", defined as the repeated exposures to the same information, substantially boosts future retention probability and reduces the rate of memory loss. Individuals who have experienced two or more learning sessions of the same item will remember more when the sessions are separated by time. The degree of memory retention directly correlates to the size of the time gap (or spacing) between sessions—the greater the spacing, the more likely that a subject will retain the

memory. For experimentally measured results and worthwhile discussions of these spacing effects, see Bahrick (1987 and 1993) [8], [9], Cepeda (2008 and 2009) [10], [11], Goverover (2009) [12], Pavlik (2003 and 2008) [13], [14], and Rohrer (2010) [15].

Bailey [16] reports a laboratory study with a learning curve in a logarithmic curve amounting to essentially the same equation as (2), albeit with different parameters ($\theta = 3.11$ and $\psi = 0.41$), with marginal time to complete the task is the dependent variable and the number of iterations it took a worker to complete the task as the dependent variable. Bailey [16] and others in the literature emphasize that the learning curve parameters are different from relearning curve or forgetting curve.

3. APPLICATION TO SOFTWARE ENGINEERING

A Work Breakdown Structure (WBS) is a hierarchical decomposition of the work activities in a software project. The lowest level activities in the WBS hierarchy are tasks. Each element of a WBS is named using a verb phrase to denote the process-oriented nature of a WBS. Another technique, the "architecture decomposition view" technique is normally used in close cooperation with the WBS technique to assemble WBS packages into larger groupings.

A WBS specifies work packages for tasks. Work packages for activities aggregate the work packages for subordinate activities and tasks. A WBS decomposes large work activities such as analysis, development, design, coding, and testing, into smaller tasks of 40 to 80 staff-hours each using the "augmented rolling wave" approach to planning. The WBS documents each task in a work package and each work package becomes a negotiated contract between the team leader and the teams or individuals assigned to that work package.

Each specific work package describes a task as follows: the order of precedence for the task. For example the magnitude of importance for that task in comparison to other activities and tasks, the planned duration of the task, a description of resources needed to accomplish the task, the work products to be produced, the risk factors associated with completing the task, and the acceptance criteria for the resulting product. Each work package must produce one or more tangible work products that satisfy some objective acceptance criterion.

The project manager estimates the time durations needed to complete each task and will negotiate resource allocation with each team member, and then assign work packages to teams or individuals subject to overall resource and scheduling constraints. Assignment of a work package from a team leader to a team member is analogous a contract for completion of one or more work products that meet an acceptance criteria within a specified time duration. By choosing the most qualified team or individual to complete a specific work package, a project manager can mitigate project risks, improve effectiveness, reduce defect levels, and speed completion.

4. EXAMPLES

How can knowledge of the Ebbinghaus [5] memory retention decay rate curve help to improve the effectiveness of project managers who allocate resources for work projects? This knowledge can help by allowing the project manager to schedule the work team with the greatest memory retention of prior programming work to specific work packages and maintenance following completion of the project. The project manager can predict the likely code retention of different teams of programmers, and make decisions based on these quantitative memory retention metrics. The conducted research provides two examples worthy of further exploration: using memory retention metrics for scheduling work packages for a large software project, and using the metrics for software maintenance by individual programmers.

A. Work Package Scheduling Example

Assume a large software development project is progressing as shown in Figure 2 below. Work packages are scheduled using the PERT analysis technique as shown in the chart, with two teams A and B performing the development of work packages 1 through 8, upon which work package number 9 depends.

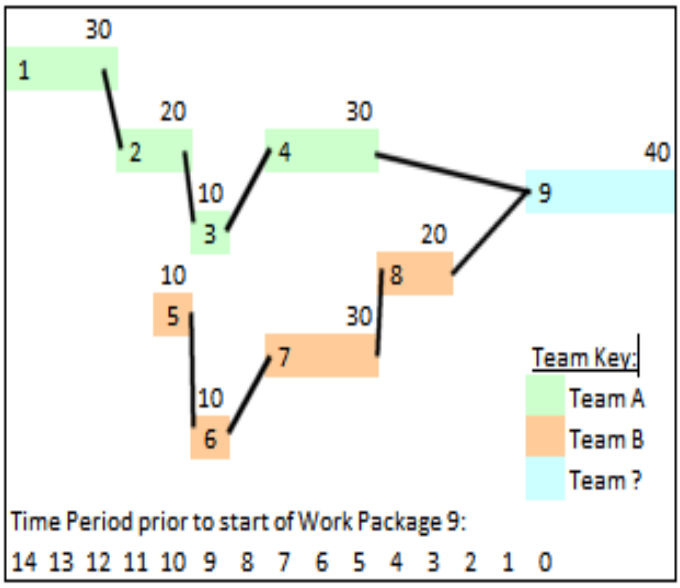

Figure 2. Sample work scheduling PERT chart.

Under our scenario, Team A has completed work packages 1 through 4, and Team B has completed work packages 5 through 8. The scheduling and completion of these various work packages is shown in the above PERT chart, as well as starting completion time in weeks prior to the scheduled start of work package number 9. The "thousands of lines of code" (KLOC) for each work package appears above the respective work packages.

A description of the work completed is shown below in Table I for work packages 1 through 8. Team A has completed 90,000 lines of code, and Team B has completed 70,000 lines of code for the various work packages. However, notice in Figure 2 that Team A's experience started at least four weeks prior to the start of Team B's work, and ended two weeks prior to the end of Team B's most recent effort on work package 8:

TABLE I. TEAM WORK PACKAGE HISTORY

Work Package	KLOC	Team	Tot. KLOC
1	30	A	
2	20	A	
3	10	A	
4	30	A	90
5	10	B	
6	10	B	
7	30	B	
8	20	B	70

This poses the question—which of these two teams should the project manager schedule to begin work on work package number 9? Team A's greater experience in terms of lines of code (90,000 versus 70,000) suggests that Team A would have a greater knowledge of the existing code that will be used to help complete work package 9. However, according to the Ebbinghaus [5] decay rate equation, it turns out Team B's knowledge of the code through memory retention is expected to be nearly double that of Team A's.

Table II outlines how the likelihood of memory retention for both Team A and Team B could be calculated. The retention rate is calculated according to the decay rate function (ψ = 1.286 and θ = 1, from Donkin, 2012 [7]), using as the time variable interval the number of periods a work package ended prior to the start of an impending work package (such as work package number 9).

This retention rate is then multiplied by the number of thousands of lines of code in the respective work package. The result is a weighted retention rate that takes into consideration the quantity of code written, as well as its recency. After adding the weighted retention rates together for each work package for both teams, Team B's weighted retention comes in at nearly double that of Team A, at 14.525 versus 8.290, respectively.

TABLE II. TEAM WEIGHTED RETENTION CALCULATION

Work Package	KLOC	Team	Periods	Retention	W. Ret.	Total
1	30	A	11	0.0458	1.373	
2	20	A	9	0.0592	1.185	
3	10	A	8	0.0689	0.689	
4	30	A	4	0.1681	5.043	8.290
5	10	B	9	0.0592	0.592	
6	10	B	8	0.0689	0.689	
7	30	B	4	0.1681	5.043	
8	20	B	2	0.4100	8.200	14.525

Although the PERT chart seems to indicate that the teams have nearly identical experience and would assumedly maintain the same memory retention, the calculated memory decay rate suggests that Team B is a much better choice in terms of current code familiarity and should therefore be more efficient at completing work package number 9 than Team A.

This result suggests that a calculated quantitative metric contrasts sharply with typical intuitive and subjective judgments. Since Team B's familiarity is nearly double that of Team A's, this degree of memory retention could result in significantly higher productivity and more efficient code development with less defects introduced due to lapses of knowledge.

B. Software Maintenance Scheduling Example

The proposed technique can be applied to software maintenance as well. Given a choice between two programmers, programmer X and programmer Y, which should be assigned to perform maintenance on a large software program? Assume programmer X was initially responsible for writing the program, although this development took place ending approximately one year, or 52 weeks, ago. Another programmer, programmer Y, was brought in relatively recently to modify 10,000 lines of code, and this project ended approximately 8 weeks ago. Also, programmer Y added 2,000 lines of new code, and this effort ended five weeks prior to the scheduled start of maintenance. At first glance, it would seem programmer X, having written 100,000 lines of code on the software, would be the better choice in terms of code familiarity. However, when applying the memory decay calculation, programmer Y actually has a better memory of the code.

The weighted retention of programmer X turns out to be 0.620, based on 52 time periods and 100,000 lines of code. The weighted retention of programmer Y is 0.942, based on the 10,000 lines of code ending eight weeks ago, and 2,000 lines of code ending five weeks ago. These results are shown below in Table III.

TABLE III. PROGRAMMER EXPERIENCE

Work	KLOC	Prog	Periods	Retention	W. Ret.	Total
1	100	X	52	0.0062	0.620	0.620
2	10	Y	8	0.0689	0.689	
3	2	Y	5	0.1262	0.252	0.942

This analysis demonstrates that programmer Y is a better choice than programmer X for maintenance work—which once again contrasts with the likely human subjective judgment as to which of the two programmers would be more familiar with the code and would perform more efficient maintenance as a result.

C. Staffing Scenario

Now, consider a staffing scenario where a new team is being put together for a software engineering project. We are now going to employ equation (2) again to estimate the learning it would take for a new team member to achieve acceptable level of performance.

Since equation (2) is an ever decreasing function, thus no one can achieve perfection; we assume that after a certain amount of practice, a team member achieves an 'optimum' stage. Suppose team member A has already achieved the 'optimum' stage. We ask, how long it is going to take a new team member B to come a certain percentage close to team member A?

We can rewrite equation (2) with time as the dependent variable:

$$t = \sqrt[\psi]{\frac{\theta}{m}} \tag{3}$$

Barring personal factors, we rewrite equation (2) for the two team members.

$$t_A = \sqrt[\psi]{\frac{\theta}{m_A}} \tag{3}$$

$$t_B = \sqrt[\psi]{\frac{\theta}{m_B}} \tag{4}$$

Thus, the time it would take for a new team member B to come to the level of A would be:

$$\Delta t = t_A - t_B = \sqrt[\psi]{\frac{\theta}{m_A}} - \sqrt[\psi]{\frac{\theta}{m_B}} \tag{5}$$

Since multiple skills are utilized in any given software project, and that equation parameters θ and ψ are going to be different for each of those skills, the combined competency level of a team member be a max function of all the Δt's at best and a summation of all the Δt's as a worse case situation.

5. CONCLUSION

A novel technique has been shown for scheduling more efficient work teams and programmers to perform software development and project maintenance. Calculating likely code retention is a relatively simple and straightforward technique using the Ebbinghaus [5] decay rate function.

The technique could be expanded to many different areas of software engineering, as well as to project management in general. Being able to calculate ahead of time how much people will remember can be useful in identifying the most efficient programmers.

This technique could possibly be combined with the aforementioned "spacing effect", which postulates that an increased length in time in between reinforcement sessions increases the likelihood of memory retention. For more on this "spacing effects" phenomenon, see the reference for the 2008 article by Cepeda [10].

6. REFERENCES

[1] Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. Psychological Science, 2, 396–408.

[2] Wickelgren, W. A. (1974). Single-trace fragility theory of memory dynamics. Memory & Cognition, 2, 775–780.

[3] Wixted, J. T., & Ebbesen, E. B. (1991). On the form of forgetting. Psychological Science, 2, 409–415.

[4] Wixted, J. T., & Carpenter, S. K. (2007). The Wickelgren power law and the Ebbinghaus savings function. Psychological Science, 18, 133–134.

[5] Ebbinghaus, H. (1913). Memory: A contribution to experimental psychology. New York: Teachers College, Columbia University. (Original work published 1885).

[6] Wixted, J.T. (2004). On common ground: Jost's (1897) law of forgetting and Ribot's (1881) law of retrograde amnesia. Psychological Review, 111, 864–879.

[7] Donkin C., & Nosofsky M. (2012). A Power-Law Model of Psychological Memory Strength in Short- and Long-Term Recognition. Psychological Science, 23(6) 625–634.

[8] Bahrick, Harry P.; Phelphs, Elizabeth. Retention of Spanish Vocabulary Over 8 Years. (1987) Journal of Experimental Psychology: Learning, Memory, and Cognition, Vol 13(2), Apr 1987, 344-349

[9] Bahrick, H. P., Bahrick, L. E., Bahrick, A. S., & Bahrick, P. E. (1993). Maintenance of foreign language vocabulary and the spacing effect. Psychological Science, 4, 316–321.

[10] Cepeda NJ, Vul E, Rohrer D, Wixted JT, Pashler H. (2008). Spacing effects in learning: a temporal ridgeline of optimal retention. Psychological Science. 2008;19(11):1095–102.

[11] Cepeda, N.J., Coburn, N., Rohrer, D., Wixted, J.T., Mozer, M.C., Pashler, H. (2009). Optimizing distributed practice: Theoretical analysis and practical implications. Experimental Psychology.

[12] Goverover Y., Arango-Lasprilla J. C., Hillary F. G., Chiaravalloti N., Deluca J. (2009). Application of the spacing effect to improve learning and memory for functional tasks in traumatic brain injury: A pilot study. American Journal of Occupational Therapy 63:543–548.

[13] Pavlik, P. I., Anderson, J. R. (2003). An ACT-R model of the spacing effect. In F. Detje, D. Dorner, & H. Schaub (Eds.), Proceedings of the Fifth International Conference of Cognitive Modeling (pp. 177–182). Bamberg, Germany: Universitats-Verlag Bamberg.

[14] Pavlik Jr., P. I., & Anderson, J. R. (2008). Using a model to compute the optimal schedule of practice. Journal of Experimental Psychology: Applied.

[15] Rohrer, D. and H. Pashler (2010). Recent Research on Human Learning Challenges Conventional Instructional Strategies. Educational Researcher, 39(5), 406

[16] Bailey, Charles D. (1989). Forgetting and the Learning Curve: A Laboratory Study, Management Science, Vol. 35, No. 3, March 1989.